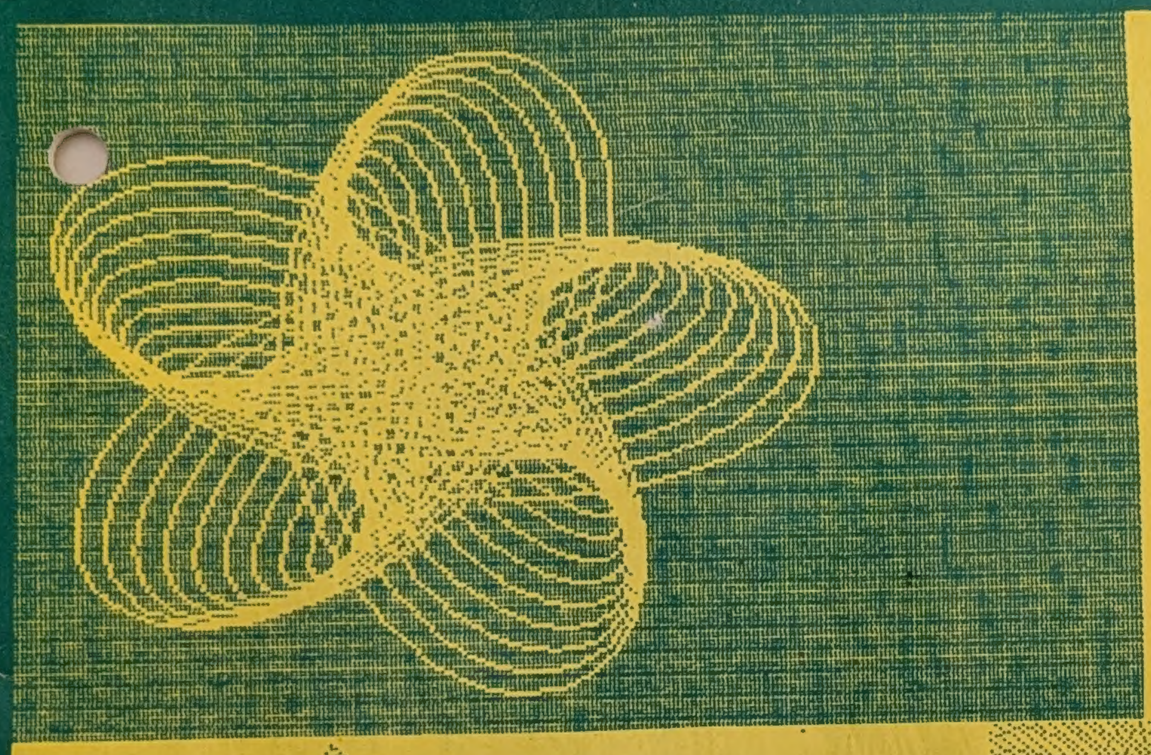
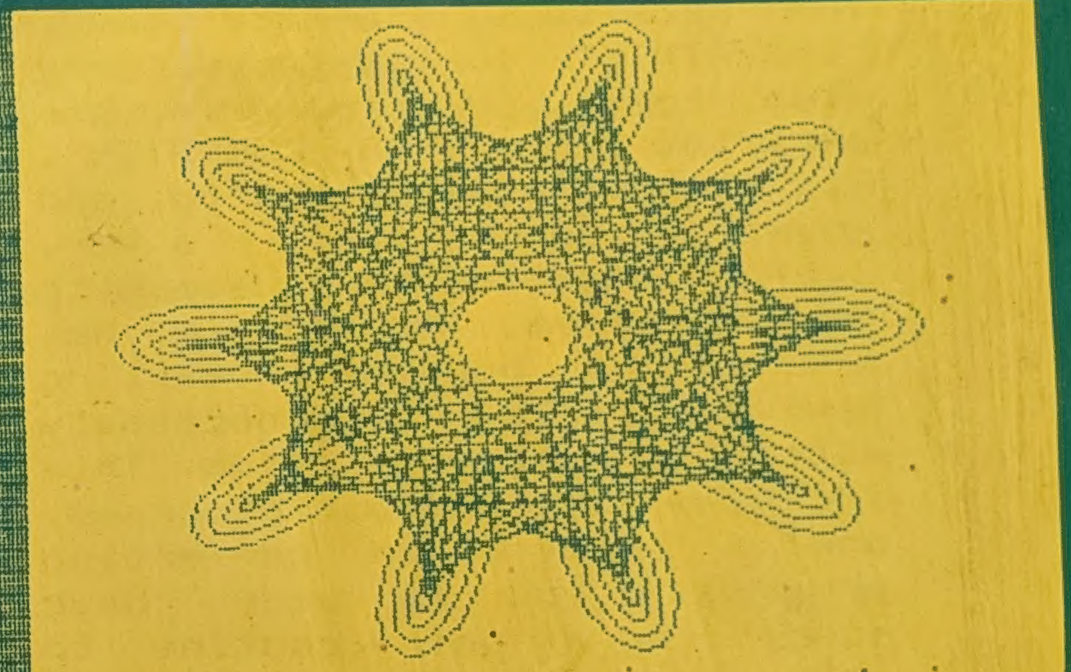
£1.00

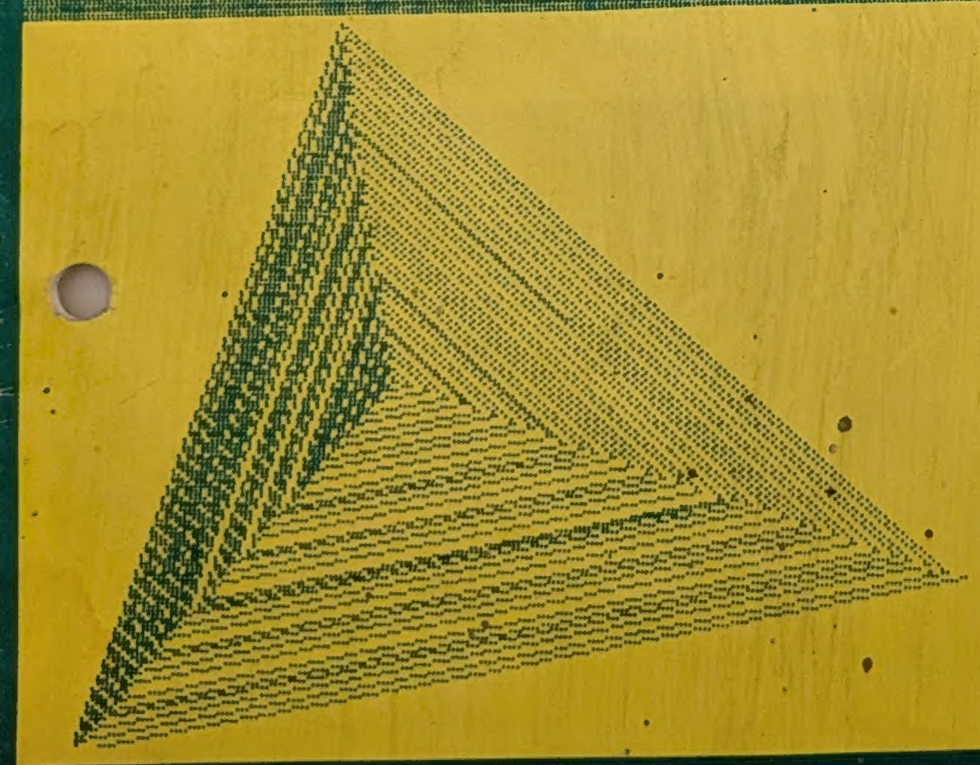
THE ATOM

No. 4

FEBRUARY 1982

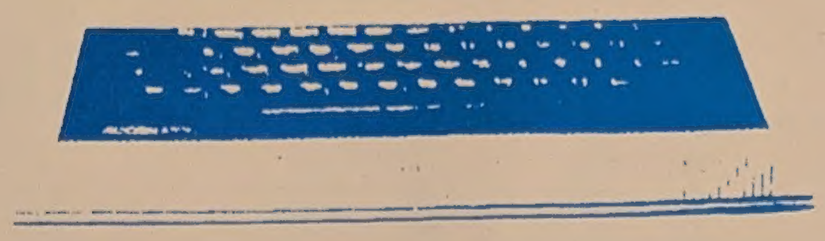






LIFE
MACHINE CODE
CLOCK

EDITORIAL



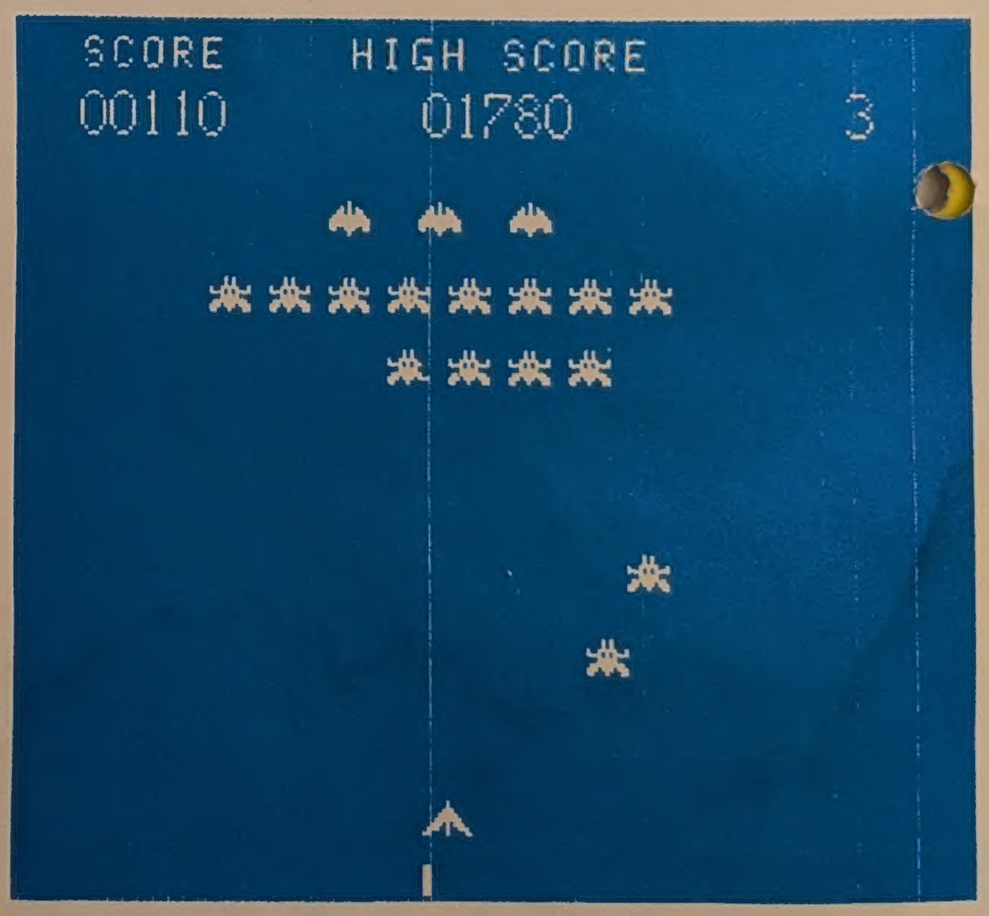
Sorry, but the gremlins got at the last issue and the description of the RACER program was printed twice. Oh well, guess you can't have too much of a good thing!

We recently heard from one of our software contributers, Mr Burgin, of Essex. He wrote the excellent 3D-maze game, "LABYRINTH" for us; you may also know of him via his articles in "YOUR COMPUTER". His latest venture is to add joysticks to his Atom - a very useful addition for use with arcade games. Mr Burgin has produced a booklet detailing how to make the necessary alterations to your Atom. This includes many sample programs, and a section on converting programs you already have (including a patch routine to allow the use of joysticks with our own "INVADERS" program). If you would like a copy of the booklet, it is available from "Burg-aids", 32 Guithavon Rd, Witham, Essex for just £2.50.

The latest release from the Bug-Byte stable is "GALAXIAN" - a superb machine code arcade game, complete with high-resolution graphics (see printout below) and sound effects. All in all, we think it's at least as good as our very popular "INVADERS". I don't know how easy it will be to convert to joystick control, but if anyone manages to get it running with Mr Burgin's joystick system, we'd be glad to pass on the patch routine.

With the imminent (hopefully!) arrival of the BBC Micro, we're considering producing a magazine for users of it, along the lines of "The Atom". Presumably, a lot of Atom users will be considering upgrading to the BBC Micro. If you are, perhaps you'd like to give us your views on the matter.Please don't swamp us with letters, but if you happen to be ordering something from us anyway, why not include a note telling us whether you'd be interested in such a magazine and if so, what sort of things you'd like to see in it.

After last issue's article on Seikosha Graphics, some kind gentleman rang us to let us know that bit 7 of the printer output is automatically available on one of the output lines tied up to the DIN socket. If so, this makes graphics printouts rather simpler than we made out last issue. Unfortunately, we've managed to lose our caller's name and the exact details, so we've had little success in trying to test out the matter. If he, or anyone else in the know would care to drop us a line, we'll have another try.



BUG-BYTE
Microcomputer Software
98-100 THE ALBANY
OLD HALL STREET
LIVERPOOL L3 9EP

© Copyright 1982

LIFE

The following, excellent version of LIFE was sent to us by Mike Wild of Harlow. It uses the full, Mode 0 screen for the display grid and runs incredibly quickly - up to 35 generations per second. Thanks are due to Mr Wild's colleague, Alec McKenzie, for help in speeding up the program to this rate. According to Mr McKenzie, the theoretical maximum for an algorithm scanning every cell on the Atom screen (with a 1MHz 6502) is about 75 gen/sec, but you'll find that you need a rather more complicated program to get much nearer to that limit. On the other hand, if you prefer to live life at a more sedate pace, try inserting a liberal sprinkling of WAIT statements into the DO, UNTIL loop at line 180.

In the initial editor, to set up the starting pattern, use keys I,J,K & M to move the cursor up, left, right and down, respectively. Key 1 sets the cell under the cursor, 0 clears it and R runs the program when the pattern is complete.

The program requires 2K of lower text space.

```
  0 REM LIFE
  1 REM (c) copyright M.J.Wild 1981
 10 DIM LL9; FOR I=0 TO 9; LLI=-1; NEXT; Q=£2800
 20 H=£80; M=H+2; L=H+4; D=H+6; C=H+8; Y=H+9; Z=H+10; S=L
 30 P.$21; GOS.a
 40 P.$21; GOS.a
 50 A=48; LI.LL0; LI.LL8; LI.Q
 60 X=0; Y=0; C=0; S=£81C1; V=£20; ?S=£FF
 70 DO
 80  LINK LL9
 90  IF C=42 IF X>0 X=X-1
100  IF C=43 IF X<29 X=X+1
110  IF C=41 IF Y<13 Y=Y+1
120  IF C=45 IF Y>0 Y=Y-1
130  IF C=17 V=£7F
140  IF C=16 V=£20
150  WAIT; ?S=V; S=£81C0-32*Y+X+1; WAIT; V=?S; ?S=£FF
160 U.(C=50)
170 ?S=V
180 DO LI.Q; U.0
190 END
200aP=Q;[
210\initialize pointers
220 LDA @£20; STA M; ASL A; STA L
230 ASL A; STA H+1; STA M+1; STA L+1; ASL A; STA H
240 LDA @£21; STA D; LDA @£82; STA D+1; LDA @£E; STA C
250\setup sums for this line
260:LL1; LDY @£1E; LDA @0; STA Z; CLC
270 ADC (H),Y; ADC (M),Y; ADC (L),Y; AND @£F
280 STA Y; BPL LL6
290\calc sum and set dest
300:LL2; INY; LDA (M),Y; DEY; BPL LL5
310:LL3; LDA @£7F; BNE LL5
320:LL4; LDA @£20;:LL5; STA (D),Y
330:LL6; LDA @0; CLC; DEY; BMI LL7
340 ADC (H),Y; ADC (M),Y; ADC (L),Y
350 AND @£F; TAX; CLC; ADC Y; ADC Z; BEQ LL4
360 PHA; LDA Y; STA Z; STX Y; PLA; AND @£F
370 CMP @£D; BEQ LL3; CMP @£C; BEQ LL2; BNE LL4
380\init for next line
390:LL7; LDA M; STA H; LDA M+1; STA H+1
400 LDA L; STA M; ADC @1; STA D; ADC @£1F; STA L
410 LDA L+1; STA M+1; STA D+1; ADC @0; STA L+1
```

```
420 INC D+1; INC D+1; DEC C; BNE LL1
430\copy back to screen
440:LL8; LDY @0
450 LDA £8200,Y; STA £8000,Y; LDA £8300,Y; STA £8100,Y
460 INY; BNE P-13; RTS
470\clear buffer to acc.
480:LL0; LDY @0
490 STA £8200,Y; STA £8300,Y; INY; BNE P-7; RTS
500\auto-repeat getkey
510:LL9; JSR £FE71; LDA @£FF; CPY £32C; STY £32C; BEQ P+7
520 STA £349; BNE P+8
530 ASL £349; BEQ P+3; TAY
540 STY £324; RTS
550]; P.$3$6; R.
```

INTERRUPT DRIVEN CLOCK

This program produces an hours, minutes and seconds printout on the screen. It uses the T1 counter of the 6522 VIA to produce regular interrupts which update the clock. This allows you to run other programs at the same time as the clock; so long as they do not interfere with the interrupts or use the same program memory. Certainly the clock will tick away quite happily whilst you are busy writing another program. Just what you need to make sure you don't get carried away and miss lunch !

As given, the program assembles into the #2800 floating point workspace; so you'll corrupt it if you use floating point variables. By changing the expression: "P=#2800" in line 10, you can relocate the program anywhere in memory. Make sure you save the assembler listings in case you wish to do this. To start the clock, RUN then LINK to the base address (LINK#2800). For quick loading, save just the machine code program itself, using:-

```
*SAVE"CLOCK"2800 289E
```

Reloading it, using *RUN"CLOCK", will run it automatically. To set the time; e.g. to 2 hours, 37 minutes, 56 seconds; type:-

```
!#AC=#02375600
```

- and hit return when this is exactly correct. If you hit BREAK, or use a cassette command, you will have to restart the clock (LINK#2800).

If the timing is out on your machine, adjust the values stored in #B804 (fine tuning) and #B807 (coarse tuning) on line 23. There is, of course, no need to type in the comments (lower case) if you're feeling lazy. If you have problems getting the program running, check that the VIA is present AND that link 2 has been made on the circuit board.

If you wish to find out more about using the 6522 VIA, get hold of the relevent data sheets from an electronic components supplier.

```
 1DIMMM7;F.Z=0TO4;MMZ=-1;N.
 5P.$21;F.Z=0TO1
10P=#2800;L=(P+#2B)%256;H=(P+#2B)/256;[
20LDA@#37;STA#B80E;LDA@#C0;STA#B80E\enable timer 1 interrupt
21LDA@L;STA#204;LDA@H;STA#205\re-vector interrupt
22LDA@#C0;STA#B80B\T1 free running mode
23LDA@#A8;STA#B804;LDA@#A2;STA#B805;STA#B807;CLI;JMP#C2B2
26:MM6\main routine
27SEI;LDA#B804;TXA;PHA;TYA;PHA\preserve registers
28INC#AC;LDA#AC;CMP@24;BCCMM7
29SED;LDX@0;LDA@#60;STA#AC\set up loop
30:MMOLDA#AD,X;CLC;ADC@1;STA#AD,X\increment count
 35CMP#AC\overflow to next byte?
 40BNEMM1\if no, then carry on
 45LDA@0;STA#AD,X;INX;CPX@2;BNEP+6;LDA@#24;STA#AC\if hours byte-
 47\then change overflow condition from 60 to 24
 50CPX@3;BNEMMO\increment next byte if any left
100:MM1\routine to print time
110CLD;LDX@0;LDY@8\set up loop
120:MM2LDA#AD,X;PHA;AND@#0F;JSRMM3\print first digit of byte
130DEY;PLA;LSRA;LSRA;LSRA;LSRA;JSRMM3\print next digit
135DEY;BEQMM5;JSRMM4\print colon
140INX;DEY;BNEMM2\loop for next byte
150:MM5\restore registers & return from interrupt
155LDA@32;STA#8017,Y;LDA@0;STA#AC
160:MM7PLA;TAY;PLA;TAX;PLA;RTI
200:MM3\print digit at #8017,Y
205BIT#B002;BMIP-3
210CLC;ADC@CH"0";STA#8017,Y;RTS
250:MM4\print colon at #8017,Y
260LDA@CH":";STA#8017,Y;RTS
 900]:N.:P.$6
2000E.
```


VIC-20
Software Wanted
20% Royalties paid
Bug-Byte, 98-100, The Albany, Old Hall St., Liverpool

ATARI 800

One only, 2-month old Atari 800 system. As new. Comprises: Atari 800 with 48K RAM, BASIC and assembler cartridges. 810 Disk Drive. I/O unit with parallel and 4 serial interfaces (RS232 and 20 mA loop). All manuals.

PRICE £980.00

BUG-BYTE
Microcomputer Software
98-100 THE ALBANY
OLD HALL STREET
LIVERPOOL L3 9EP

ATOM BUSINESS

Suitable for the expanded Atom with floating point ROM

Each section contains:
— Management Summary
— Operating Instructions
— Program notes
— Source listing

SALES GRAPH
SALES RECORDS
NOMINAL LEDGER
QUEUEING
SIMULATION
EXPENSE CLAIMS

ADDLIST
LABEL PRODUCTION
LEASE OR BUY DCF
METRIC CONVERSION
STANDARD DEVIATION
BUDGET FACTORING

110 Page Book 6.95

Cassette & Book 15.57

PHIPPS ASSOCIATES
3 DOWNS AVENUE EPSOM Surrey KT18 5HG
Tel: Epsom (03727) 21215 quoting your credit card reference. 24 Hour phone service. Air Mail delivery in Europe add 50p. air mail elsewhere add £1.50 per book. Prices shown above include UK postage and VAT on cassettes

MACHINE CODE

If you can remember back to issue 2, we had got to the following stage with this short BASIC program in the normal, #29 text space:-

```
 10CLEAR2
 15MOVE0,0;DRAW127,0;MOVE0,88;DRAW127,88
 20?£80=15;?£81=£83
 30?£82=13;?£83=£83;?£84=0;?£85=£83;?£86=2;?£87=£83
100LI.£2D00
1000E.
```

-and in text space #86, the following assembler listings:-

```
 10DIMMM64;F.Z=0TO64;MMZ=-1;N.
 20F.Z=0TO1;P=£2D00
 30P.$21
 40[
 50JSRMM0
 60:MM7JSRMM0;JSRMM2
 65JSRMM8
 70JSRMM0
400JMPMM7
1050:MM0LDY@0;:MM1LDA(£80),Y;EOR@£18;STA(£80),Y
1052LDA(£82),Y;EOR@£18;STA(£82),Y
1054LDA(£84),Y;EOR@£18;STA(£84),Y
1056LDA(£86),Y;EOR@£18;STA(£86),Y
1060TYA;CLC;ADC@16;TAY;CMP@128;BNEMM1;RTS
1100:MM2LDA@£76;STA£B000;LDA£B001;CMP@253;BNEP+5;JSRMM3
1110LDA£B002;AND@£40;BNEP+5;JSRMM4
1120LDA@£72;STA£B000;LDA£B001;CMP@254;BNEP+5;JSRMM5
1130LDA@£74;STA£B000;LDA£B001;CMP@254;BNEP+5;JSRMM6;RTS
1140:MM3SEC;LDA£80;SBC@16;STA£80;LDA£81;SBC@0;STA£81
1150SEC;LDA£82;SBC@16;STA£82;LDA£83;SBC@0;STA£83;RTS
1160:MM5SEC;LDA£84;SBC@16;STA£84;LDA£85;SBC@0;STA£85
1170SEC;LDA£86;SBC@16;STA£86;LDA£87;SBC@0;STA£87;RTS
1180:MM4CLC;LDA£80;ADC@16;STA£80;LDA£81;ADC@0;STA£81
1190CLC;LDA£82;ADC@16;STA£82;LDA£83;ADC@0;STA£83;RTS
1200:MM6CLC;LDA£84;ADC@16;STA£84;LDA£85;ADC@0;STA£85
1210CLC;LDA£86;ADC@16;STA£86;LDA£87;ADC@0;STA£87;RTS
1300:MM8LDY@0;LDA(£80),Y;AND@£18;BEQP+5;JSRMM4
1310LDA(£84),Y;AND@£18;BEQP+5;JSRMM6
1320LDY@112;LDA(£80),Y;AND@£18;BEQP+5;JSRMM3
1330LDA(£84),Y;AND@£18;BEQP+5;JSRMM5;RTS
10000];N.;P.$6;E.
```

If you have these programs on tape, load them in (using *LOAD to get them into the correct text spaces). Running the assembler listings program and then the BASIC gives a screen with four bats which move rapidly under keyboard control. However, the action is hard to see due to a severe attack of screen interference. This is caused by writing to the screen memory whilst the TV screen is being refreshed. To avoid the problem, all use of the screen memory must be done during the TV "flyback time" - this is signalled by bit 7 of address #B002 being a zero. Add the following to the listings:-

```
390LDA#B002;BMIP-3
```

The BMI instruction is passed over if bit 7 of #B002 is zero and the program continues as usual. If bit 7 is not zero, the TV screen is being accessed and the program branches back to the LDA#B002 instruction to test bit 7 again. Eventually flyback will occur and the program will continue. RUN the listings and then the BASIC to see the effect.

Makes quite a difference doesn't it? Now all the "snow" has disappeared from the screen and the bats move at a more reasonable pace. At present, by a bit of luck, they're moving at about the right speed. If they'd still been too fast, we'd have had to put some sort of delay loop into the main loop of the program to slow them down a bit. If, on the other hand, they'd moved too slowly, we'd have had to move them 2 spaces at a time rather than one (replacing all the 16's in lines 1140 to 1210 with 32's). For the moment we'll leave well alone, but they may need speeding up later if the rest of the program slows down the main loop.

At this stage of development, things are beginning to build up to a fair degree of complexity and you're liable to start forgetting what the various subroutines do and what we've used various zero page locations for. This can lead to bugs which are very difficult to track down. Imagine using location #80 to keep track of the score, forgetting that it was also being used to keep track of the position of one of the bats. All though your program might seem faultless, the score would change as the bats moved, a bat would jump around as the score went up and generally all hell would be let loose. For the sake of your future sanity, keep a brief note of what each subroutine and zero page location does. e.g.:-

MM0 - Print/delete bats
MM2 - Scan keys and move bats
MM3 - Move player 1's bats up
MM4 - Move player 1's bats down
MM5 - Move player 2's bats up
MM6 - Move player 2's bats down
MM7 - Main loop
MM8 - Check for bat collisions

#80,1 - Player 1 bat 1 address
#82,3 - Player 1 bat 2 address
#84,5 - Player 2 bat 1 address
#86,7 - Player 2 bat 2 address

Up-date your list as we continue with the program.

Referring to our original plan of action, the next obvious step is to get the ball into the picture and moving. As with the bats, we'll need a couple of memory locations to store a base address (#88,9). The bat will consist of a square of four dots, so to plot this we'll also need a third memory location, with 2 bits set and the rest cleared, to act as a "bit mask". With the bats, the immediate data #18 served this purpose. The ball must be free to move sideways across the screen so its bit mask will vary. e.g. With the ball moving left to right :-

BIT PATTERN	HEX VALUE
11000000	#C0
00110000	#30
00001100	#0C
00000011	#03

Location #8A will be used to store the changing value of the bit mask. The mask needs to be exclusive-OR'd with the ball's screen location (pointed to by #88,9) and stored there. The same is done with the location 16 bytes on for the lower half of the ball:-

```
1400:MM9LDY@0;MM10LDA(#88),Y;EOR#8A;STA(#88),Y
1410TYA;CLC;ADC@16;TAY;CMP@32;BNEMM10;RTS
```

Note the similarity to the bat printing routine (lines 1050 and 1060). Call the routine at the start of the program:-

```
52JSRMM9
```

RUN the program to assemble it and then BREAK, OLD, LIST to get to the BASIC program. Set up some starting values for the ball:-

```
40?#88=8;?#89=#83;?#8A=#30
```

RUN the BASIC.This should put the ball somewhere near the middle of the screen.Now to move it! Up and down's easy - just like the bat:-

```
1500:MM10SEC;LDA#88;SBC@16;STA#88;LDA#89;SBC@0;STA#89;RTS
1510:MM11CLC;LDA#88;ADC@16;STA#88;LDA#89;ADC@0;STA#89;RTS
```

To move to the left, we just shift left the bits in the bit mask:-

```
1520:MM12ASL#8A;ASL#8A
```

- Twice to make sure the ball's top speed is fast enough. Checking the carry flag lets you know if you've shifted the bits right out of the byte. If so, reset the bit mask and decrease the base address:-

```
1530BCSP+3;RTS;DEC#88;LDA@3;STA#8A;RTS
```

And to move right:-

```
1540:MM13LSR#8A;LSR#8A;BCSP+3;RTS;INC#88;LDA@#C0;STA#8A;RTS
```

The speed and direction in which the ball will move depends on the frequency with which these routines are called. Let's concentrate on the left-right motion first. To start with we'll use location #8B to represent whether the ball's moving left or right - store 0 in it for left, 1 for right. Location #8C we'll use to represent the speed of the ball - 1 for top speed, 255 for slowest. Location #8D will be decremented at regular intervals. When it becomes zero it is reloaded with the value in #8C and the appropriate movement routine is called:-

```
1550:MM14DEC#8D;BEQP+3;RTS;LDA#8C;STA#8D
1560LDA#8B;BNEP+5;JMPMM12;JMPMM13
```

Now a quick try out. Call this routine from the main loop:-

```
80JSRMM14
```

And remembering to rub out the ball before moving it and re-plotting it afterwards, to prevent it spreading all over the screen, edit line 60 to:-

```
60:MM7JSRMM9;JSRMM0;JSRMM2
```

And add:-

```
100JSRMM9
```

RUN the listings then get back to the BASIC program and set variables to move the ball left at top speed:-

```
50?#8B=0;?#8C=1;?#8D=1
```

Then RUN. The ball behaves a bit oddly when it goes off the edge of the screen, but in the final version, we won't let it do that.

BREAK and change line 50 to:-

```
50?#8B=1;?#8C=255;?#8D=255
```

RUN and the ball moves right at its slowest speed. Try out a few other values in line 50. You should find that there's a wide enough range of speeds to suit all needs.

Now a similar routine for the vertical motion, using #8E,8F and 90 as variables:-

```
1570:MM15DEC#90;BEQP+3;RTS;LDA#8F;STA#90
1580LDA#8E;BNEP+5;JMPMM10;JMPMM11
```

Link this into the main loop with:-

```
90JSRMM15
```

Best not to run this yet because there's nothing to stop the ball shooting off the bottom of the screen and through your listings, causing havoc as it goes! We need a routine to test if there's anything under the ball before it's plotted:-

```
1600:MM16LDY@0;LDA(#88),Y;AND#8A;BEQP+3;RTS;LDY@16;LDA(#88),Y
1610AND#8A;RTS
```

On return from this routine, the zero flag will be cleared if there is an obstruction under the ball. So if we call this routine after moving the ball up and find the zero flag cleared, we must change the ball's direction (in #8E) and immediately move the ball back the other way:-

```
92JSRMM16;BEQP+15;LDA@1;EOR#8E;STA#8E;LDA@1;STA#90;JSRMM15
```

The BEQP+15 jumps to the end of the line if the ball has a free path.

Similarly, for the left-right motion:-

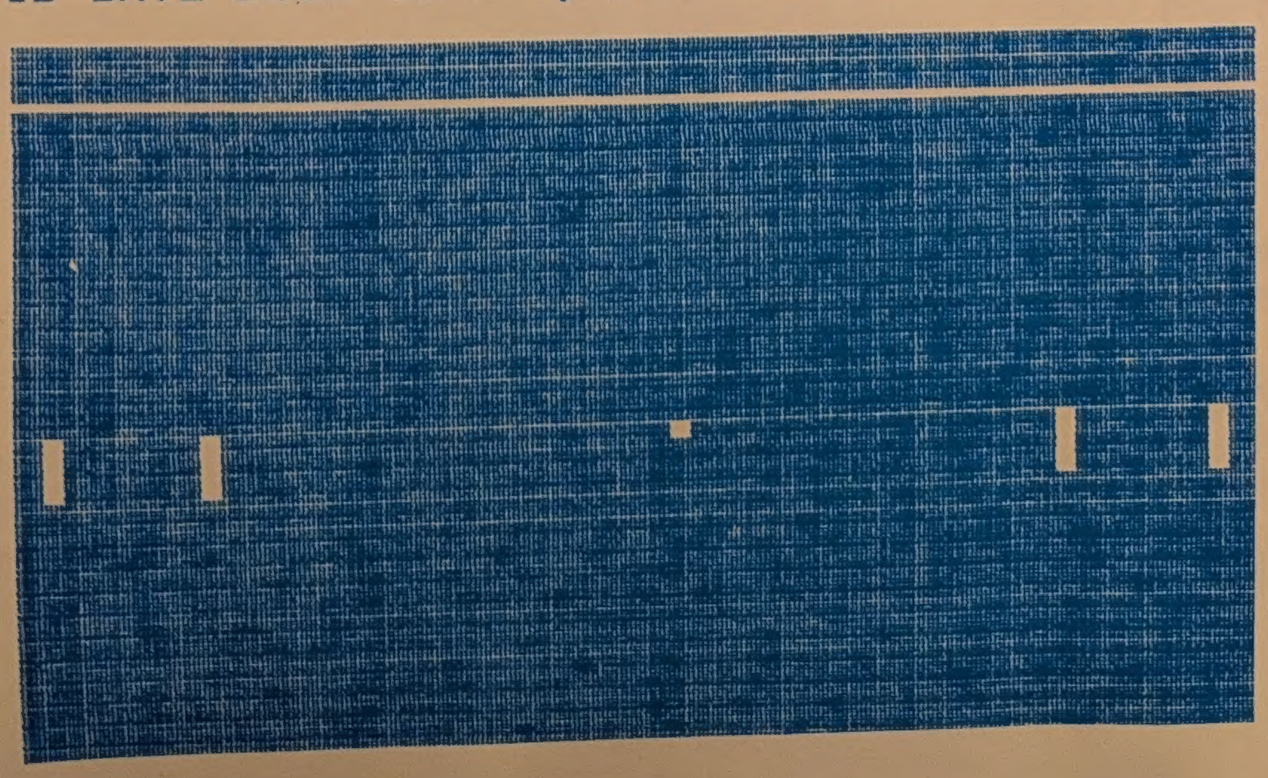
```
82JSRMM16;BEQP+15;LDA@1;EOR#8B;STA#8B;LDA@1;STA#8D;JSRMM14
```

Now RUN the listings, get back to the BASIC program and edit in:-

```
50?#8B=1;?#8C=2;?#8D=2
60?#8E=1;?#8F=2;?#90=2
```

Now RUN. The ball now bounces off the sidewalls correctly, but the bounce from the bats isn't quite correct and there are still problems when the ball goes off the sides of the screen. Try different set up values in BASIC lines 50 and 60.

Well, that's all for now. Before you get fed up and switch off, do remember to SAVE both text spaces ready to carry on.



PELMANISM

This game, from Eric Gower, of Southampton is a test of memory, for two players. It requires just over 4K of memory, but if you can live without the helpful REM statements and abbreviate as you go along, you should be able get down to well under 4K and save your fingers a little. All is explained in the instructions, from line 740 onwards.

```
1REM atom pelmanism
2REM (C) ERIC GOWER,SOUTHAMPTON
8REM************************
9REM CONTROL ROUTINE
10GOSUBz;DIMP-1;GOSUBa
15C=0;GOSUBk;GOSUBs
20GOSUBi
25GOSUBm;GOSUBn
26GOSUB1;T=1;GOSUBq;T=2
30DOT=3-T;
35GOSUBq
40GOSUBe;IFZ=0GOSUBg
45IFS=0;FORJ=1TO100;WAIT;NEXT;GOSUBh
50GOSUB1;GOSUBr
60UNTILSS1+SS2=20
65GOSUBu
70I=1;IFSS1<SS2;I=2
72IFSS1=SS2;GOTO80
75P.''''$AAI'"IS THE WINNER. WELL DONE!";G.90
80P.''''"                    A DRAW!"''"    WELL DONE BOTH OF YOU!"
85?#E1=#80
90END
98REM************************
99REM KEYBOARD SCAN
100aP.$21;[:LLOJSR#FFE3;STA#80;RTS;];P.$6;RETURN
108REM************************
109REM CHOOSE COORDDINATES
110bLINKLLO
120IF?#80>68OR?#80<65GOTOb
130L=(?#80)-65;GOSUBo
140LINKLLO
150IF?#80>57OR?#80<48GOTO140
160M=(?#80)-48;GOSUBp
170RETURN
178REM************************
179REM POSITION ON SCREEN
180cX=#7FC2+32+(L+1)*96+(M*3);RETURN
188REM************************
189REM POSITION IN ARRAY
190dN=L*10+M+1;RETURN
198REM************************
199REM CHOOSE TWO CARDS
200eZ=0;FORI=1TO2
210GOSUBb
220GOSUBc;XXI=X
230GOSUBd;NNI=N
240GGI=CCN
250IFCCN=255ANDI=1;S=1;I=2;Z=1;GOTO270
255IFCCN=255ANDI=2;?XX1=#7F;S=1;Z=1;GOTO270
260?X=CCN
270NEXT;RETURN
278REM************************
279REM CHECK CARDS ARE SAME
280gS=0;IFGG1<>GG2GOTO310
290CC(NN1)=255;CC(NN2)=255;?XX1=64;?XX2=64;S=1
```

```
300SST=SST+1;T=3-T
310RETURN
318REM************************
319REM TURN 2 CARDS FACE DOWN
320h?XX1=#7F;?XX2=#7F;RETURN
328REM************************
329REM INPUT  PLAYERS NAMES
330iFORI=1TO2
340P.?"ENTER THE NAME OF PLAYER"I" AND  PRESS return"?
345INPUT$A
350IFLENA>10A?10=13
360IF?A=13GOTO340
370$AAI=A
380NEXT;CLEARO;RETURN
388REM************************
389REM TURN CURSOR OFF
390k?#E1=0;RETURN
398REM************************
400REM PRINT NAMES
4101PRINT$30;C=0;GOSUBk
420FORI=1TO14;PRINT?;NEXT
430DOPRINT" ";UNTILCOUNT=3
440PRINT$AA1,SS1
450DOPRINT" ";UNTILCOUNT=19
460PRINT$AA2,SS2
470PRINT$13
480RETURN
488REM************************
489REM SHUFFLE CARDS
490mFORI=1TO20
500CC(2*I-1)=I;CC(2*I)=I
510NEXT
520FORI=40TO1STEP-1
530R=ABSRND%I+1
540W=CCI
550CCI=CCR
560CCR=W
570NEXT
580RETURN
588REM************************
589REM DRAW 40 CARDS
590nFORI=0TO3
600B=#7FC2+32+(I+1)*96;?(B-2)=I+1;B?29=I+1
610FORJ=0TO9
620X=B+J*3
640?(X-33)=#41;?(X-1)=#55;?(X+31)=#50
650?(X-32)=#43;?X=#7F;?(X+32)=#70
660?(X-31)=#42;?(X+1)=#6A;?(X+33)=#60
670N.;N.;FORI=0TO9;?(B+64+I*3)=I+#30;?(B-11*32+I*3)=I+#30;NEXT
673RETURN
678REM************************
679REM WRITE COORDINATES
680o?(BBT+3*I-3)=L+1;RETURN
690p?(BBT+3*I-2)=M+48;RETURN
700RETURN
708REM************************
709REM FLASH "GO" REMINDER
710qFORI=1TO2;L=#7F-1;GOSUBo;M=#7F-48;GOSUBp;NEXT;RETURN
720rFORI=1TO2;L=#40-1;GOSUBo;M=#40-48;GOSUBp;NEXT;RETURN
728REM************************
729REMINSTRUCTIONS
730sGOSUBu;P.???"DO YOU WANT INSTRUCTIONS?"?"(TYPE Y OR N)"
```

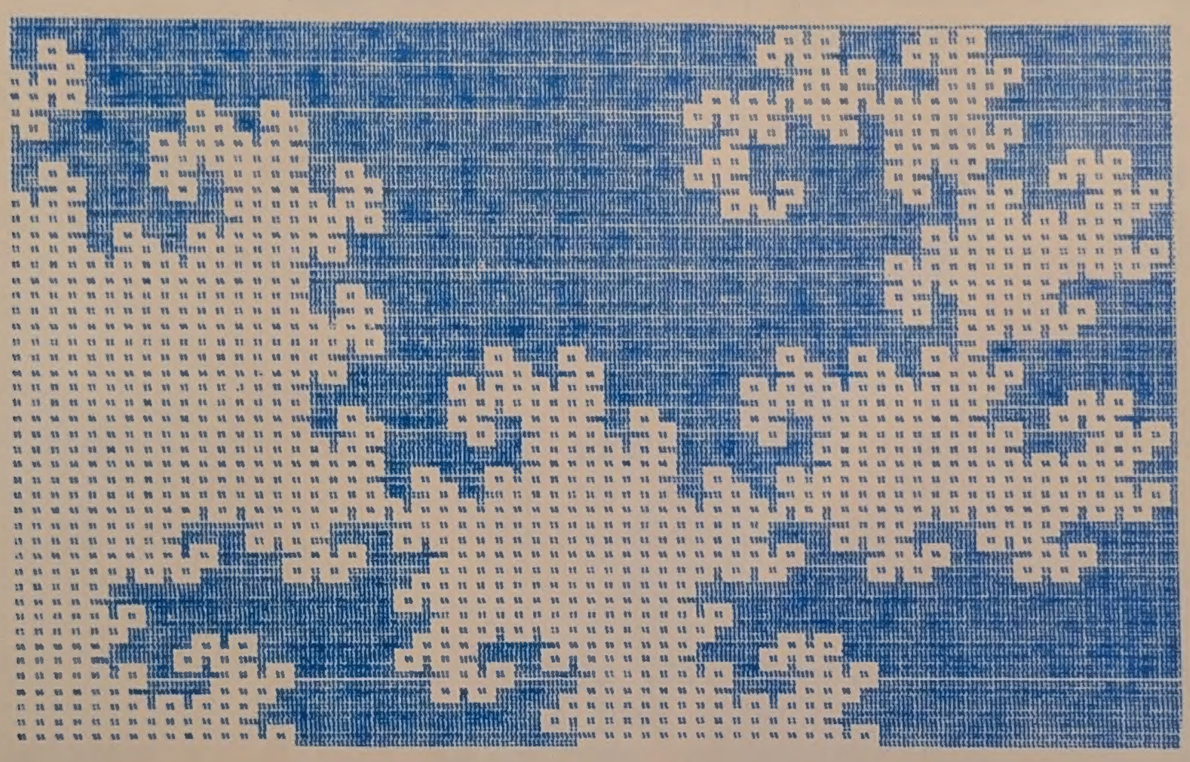
```
732LINKLLO;IF?#80<>CH"Y"RETURN
735GOSUBu
740P."  THIS GAME (SOMETIMES CALLED"'
750P."'PAIRS') IS FOR TWO PLAYERS."'
760P."THE NAMES OF THE TWO PLAYERS ARE"
770P."ASKED FOR AT THE START OF THE"'
780P."GAME."''
790P."  THEN FORTY 'CARDS' WILL BE"'
800P."SHOWN FACE-DOWN LIKE THIS:"
810GOSUBt;CLEARO;GOSUBn;GOSUBt
820GOSUBu;P.'''"  CARDS ARE CHOSEN BY ENTERING"'
830P."THEIR COORDINATES USING THE "'
840P."LETTER FIRST, E.G. A9 NOT 9A."'
850P."  THE LETTER ON THE FRONT OF"'
860P."THE CHOSEN CARD THEN APPEARS."
870GOSUBt
880GOSUBu;P.'''"PLAYERS TAKE IT IN TURN TO"'
890P."CHOOSE TWO CARDS. IF THESE MATCH"
900P."THEN THE PLAYER'S SCORE GOES UP"'
910P."AND THE PLAYER GETS AN EXTRA "'
920P."TURN."
930GOSUBt
940GOSUBu;P.'''"  PLAY CONTINUES LIKE THIS UNTIL"
950P."ALL THE CARDS HAVE BEEN PAIRED."'
960P."  CHOOSING A CARD WHICH HAS"'
970P."ALREADY BEEN PAIRED MEANS YOU"'
980P."MISS YOUR TURN.";GOSUBt
990GOSUBu;P.'''"THE GAME NOW BEGINS"'';RETURN
998REM**********************
999REM RECURRING PAGING DEVICE
1000tGOSUBk;P.$30;GOSUBk
1010P.$30'''''''''''''''"          PRESS return TO CONTINUE"
1020LINKLLO;IF?#80<>13GOTO1020
1030RETURN
1038REM**********************
1039REM  TITLE
1040uP.$12;CLEARO;P."          atom"$#80"pelmanism"'';RETURN
1998REM******************
1999REM VARIABLES AND ARRAYS
2000zDIMCC(40),XX(2),NN(2),GG(2),SS(2),A(64),BB(2),LLO
2010DIMAA(2);FORI=1TO2;DIMK(10);AAI=K;NEXT
2020T=2;SS1=0;SS2=0;Z=0;@=2
2030BB1=#8000+15*32+3
2040BB2=BB1+16
2050L=#7F-1;M=#7F-48
2060RETURN
```

DRAGON DESIGN

This program provides a good demonstration of the use of the EPSON printout routine. Load the printout routine into memory, starting at #2800, before running the program. You will have to wait about 3 minutes before the program gets round to printing, so be patient.

If you do not want a printout of the screen, remove line 155. The program will run as before but only produce its design on the screen.

```
 10 REM    Dragon Designs - see Coxeter & Rouse Ball
 20 REM    "Mathematical Recreations and Essays" 12th Edition
 30 REM    p.266 ff (Toronto, 1974)
 35 REM    Program by Michael Wild  -  29/7/81
 40 D=2; REM Initial direction
 50 L=2; REM Segment Length
 60 CLEAR 2; MOVE (21*L+1),(10*L+1)
 70 FOR N=1 TO 8192
 80  IF D=0 PLOT 1,L,0; G.b
 90  IF D=1 PLOT 1,0,-L; G.b
100  IF D=2 PLOT 1,-L,0; G.b
110  IF D=3 PLOT 1,0,L; G.b
120b M=N
130a IF (M%2)=0 M=M/2; G.a
140  D=(D+M)%4
150 NEXT N
155 X=1;LINK#2800
160 END
```



EPSON SCREEN PRINTOUT

Following last issue's article on producing a high-resolution printout using the Seikosha printer, we have a program from Mike Wild of Harlow which does the same job for those of you with an EPSON MX-70. The program also works equally well with the type II MX-80 (a fact we were rather dismayed to find out, having spent an afternoon writing a similar program for our MX-80 whilst the program below gathered dust on a shelf!).

The program transfers the contents of the Atom screen, in graphics modes 1-4, to an EPSON printer, either in normal or inverted form. The graphics mode in use is determined by examining the 8255 PPI which controls the 6847 graphics chip. If any other than modes 1,2,3 or 4 are in use, no printing is done.

As given, the program assembles into the #2800 floating point workspace. After assembly (no need to type in the lower case comment lines) the code can be saved using:-

```
*SAVE"SCREEN DUMP"2800 2900
```

With this code present in memory, having produced the required screen display, your program can call the print routine with LINK#2800. If X=0 before linking, a normal printout will result. Any other value in X will give an inverted printout.

The printouts around this article were produced using this program. See also the "DRAGON DESIGNS" program for an example of its use.

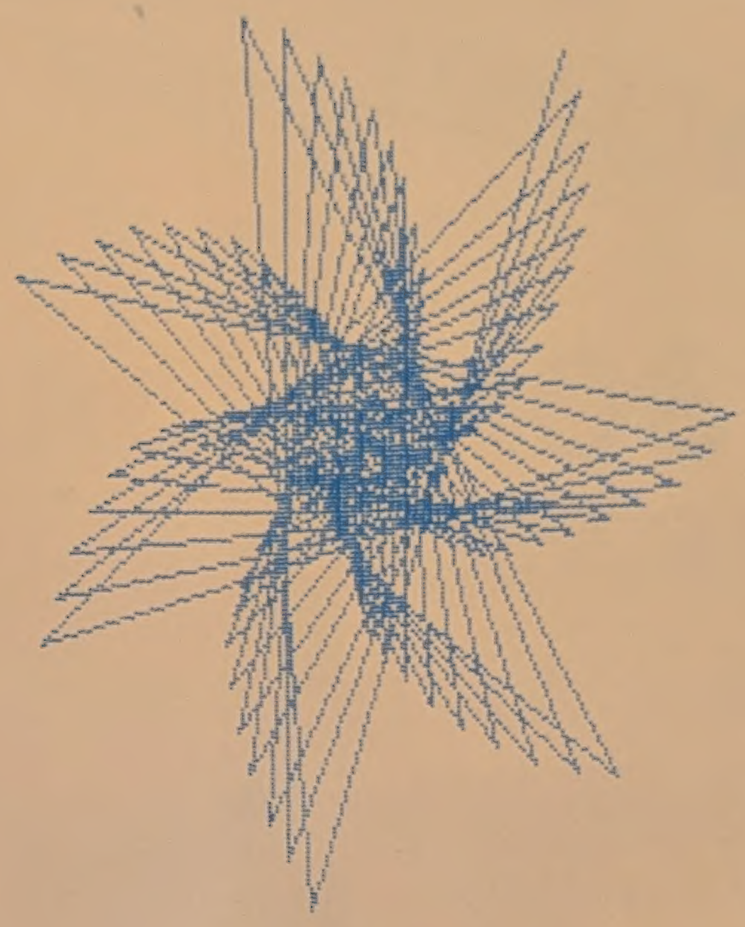
```
10 REM ********************
20 REM * SCRDUMP - V1.0 *
25 REM * BY MIKE WILD   *
30 REM * 4-8-81         *
40 REM *----------------*
50 REM * X=0  -> NORMAL *
60 REM * X<>0 -> INVERT *
70 REM ********************
100 DIM LL5,DD1; Q=£2800
110 FOR I=0 TO 7; LLI=-1; N.
120 D=£80
130 P.$21  ; GOS.A
140 P.$21  ; GOS.A
150 P.$21  ; GOS.A
999 END
1000aP=Q;[
1003\ Get mode from PPI; LDA £B000; ROL A; ROL A; ROL A
1005 BCS P+3; RTS; BPL P-1; AND @3; STA D+3
1007\ Fix invert mask; CPX @0; BEQ P+4; LDX @£3F; STX D+2
1010\ Set base; LDA @£80; STA D+1; LDA @0; STA D; PHA
1015\ Set L-cnt, LP on; LDA @£20; STA D+6; LDA @2; JSR £FEFB
1020\ Set LP, R-cnt;;LL4; LDY @6; JSR LL1; LDX D+3; LDA DD1,X
1040\ Save R-cnt, set mask;;LL5; PHA; LDA @£80; STA D+4
1080\ Init image, offset;;LL2; LDA @0; PHA; TAY
1084\ Get screen bit;;LL3; LDX @0; LDA (D),Y; BIT D+4
1086 BEQ P+3; INX; STX D+5
1088\ Set image bits; PLA; LDX D+3; CPX @1; BEQ P+7; BPL P+8
1090 ASL A; ORA D+5; ASL A; ORA D+5; ASL A; ORA D+5; PHA
1100\ New ofset; TYA; CLC; ADC DD1,X; TAY; CMP DD1+4,X; BCC LL3
1110\ Mask & o/p; PLA; EOR D+2; JSR £FF10
1115\ Twice?; CPX @3; BEQ P+5; JSR £FF10
1120\ Shift mask & go down; LSR D+4; BCC LL2
1140\ Next column; INC D; PLA; SBC @1; BNE LL5
1180\ Next line; CLC; PLA; ADC DD1+4,X; BCC P+4; INC D+1; STA D
1185\ Save R-cnt, do cr/lf; PHA; LDY @9; JSR LL1
1187\ Check esc; LDA £B001; AND @£20; BEQ P+6; DEC D+6; BNE LL4
1190\ Repair stk, LP off; PLA; LDA @3; JMP £FEFB
8000\ O/P LP command seqs
8050;LL1; LDA DD0,Y; DEY; JSR £FF10; BPL LL1; RTS
9000\ Y=6 - 6/72 spacing & 256 bitmaps
9005\ Y=9 - C/R & Normal spacing
9010;DD0;]
9015 !P=£1B4B0081;P=P+4;!P=£B21B4106;P=P+4;!P=£00000D1B;P=P+2;[
9020;DD1;]
9025 !P=£20101010;P=P+4;!P=£C0603020;P=P+4
9999 P.$3$6;R.
```

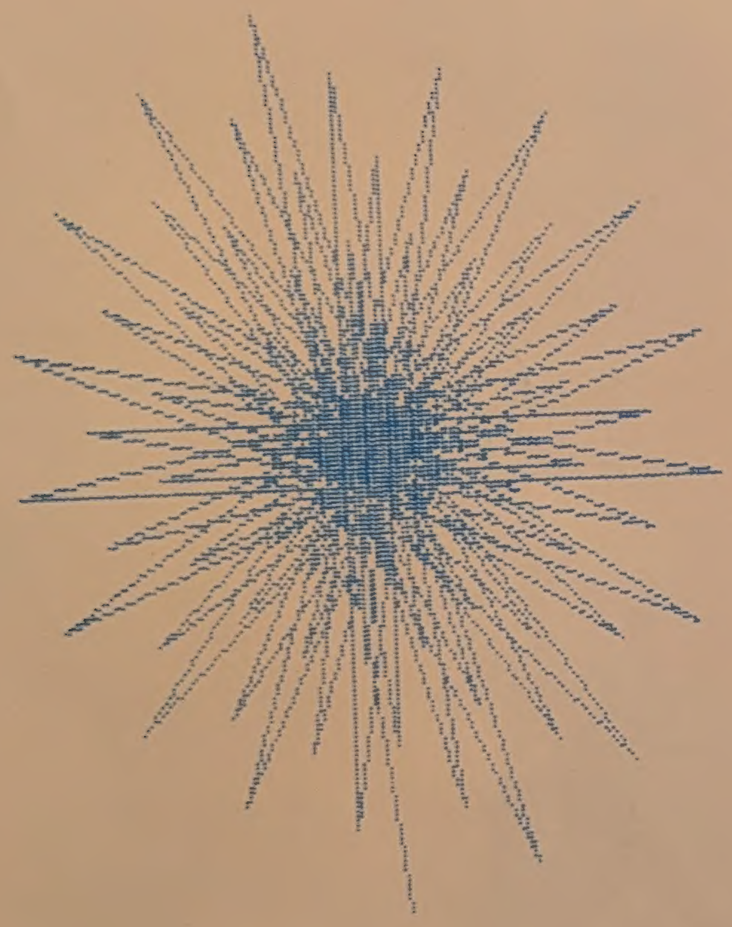
N.B. PLEASE READ £ AS #

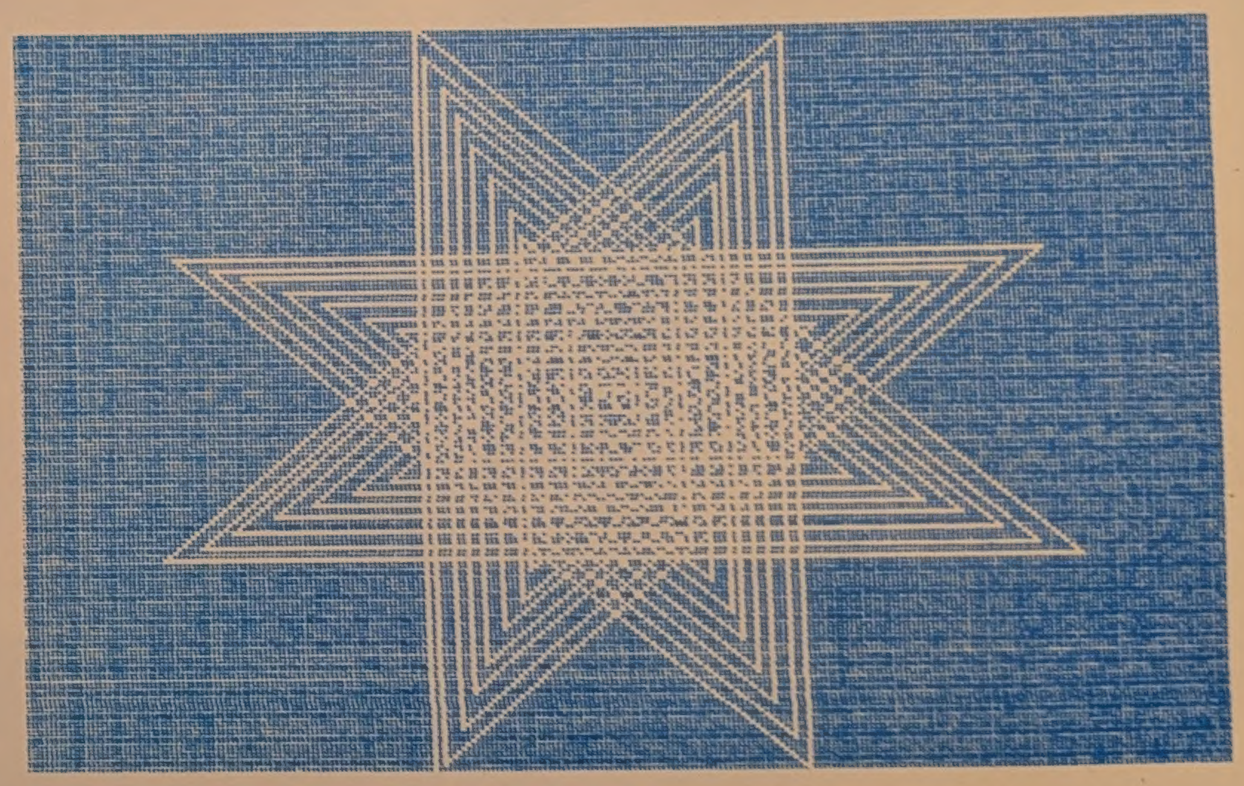
Don't forget your

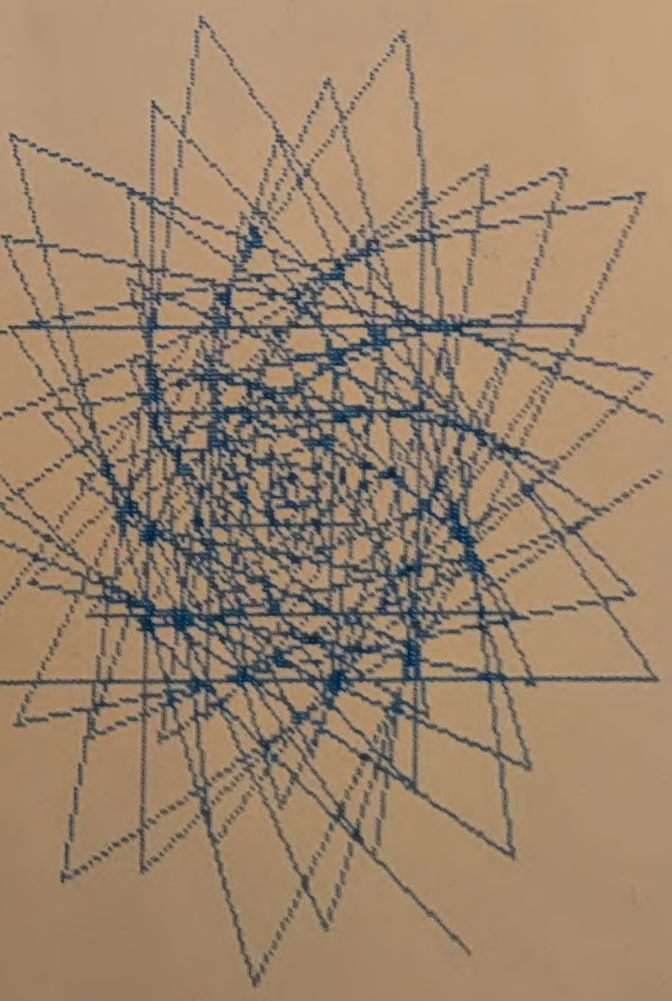
10 %

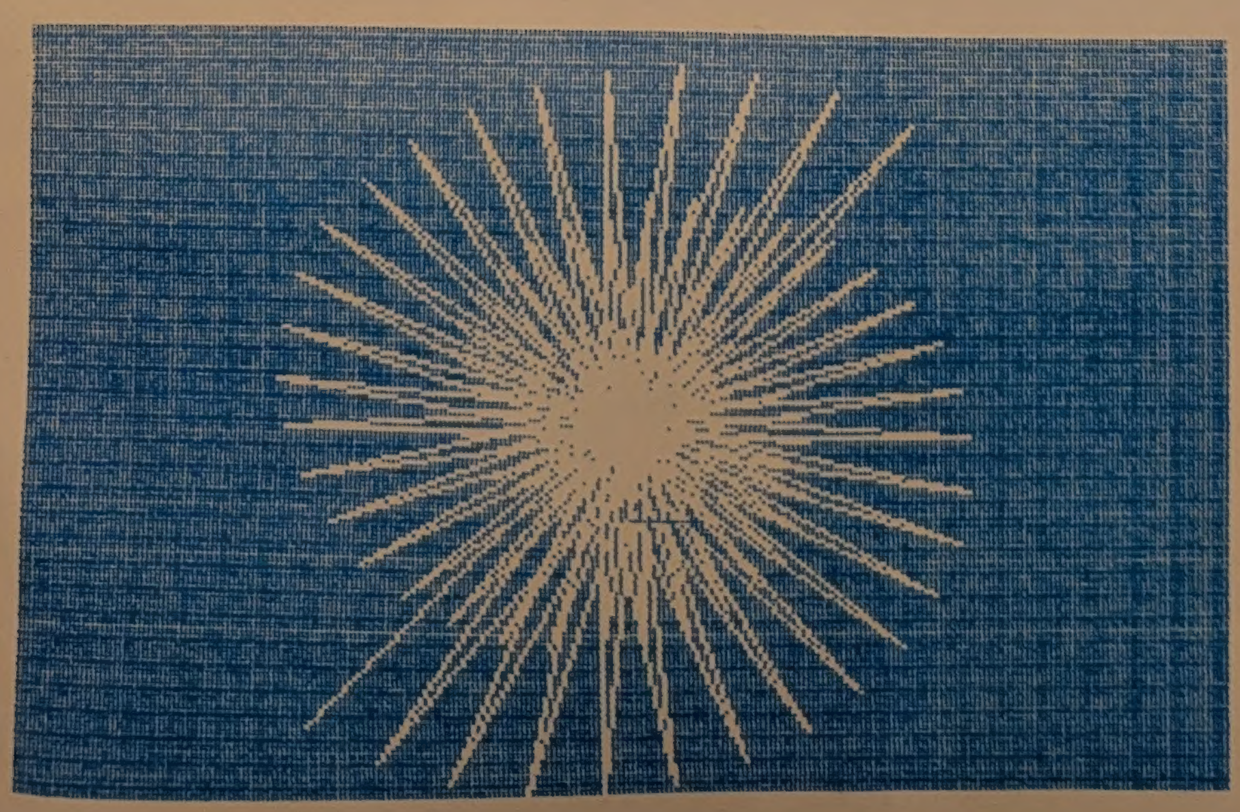
software discount









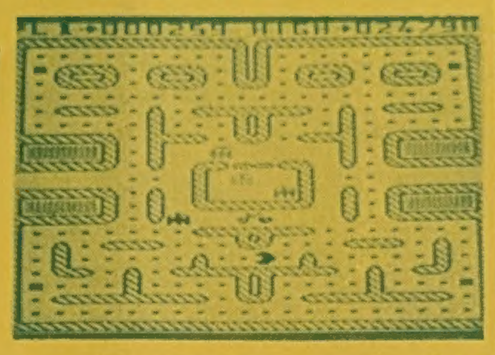


AT LAST!

ALL PRICES INCLUSIVE
TELEPHONE 051-227 2642

VIC SOFTWARE

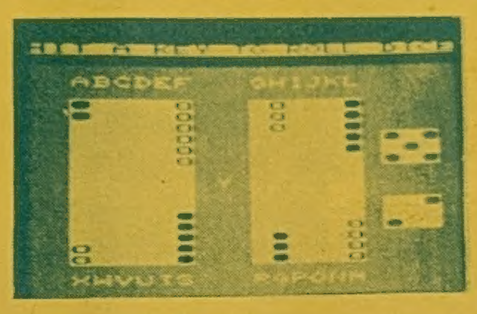
VICMEN



A full-feature version of the popular "Puckman" arcade game for the UNEXPANDED VIC. Written entirely in machine-code for fast action.

SUPPLIED ON CASSETTE at £7.00

VICGAMMON



Standard Backgammon game for the VIC with 3K expansion. Rapid computer responses. Instructions on how to play are included.

SUPPLIED ON CASSETTE at £7.00

More VIC software will be available by the time you read this. Phone 051-227 2642 for details. Generous DEALER DISCOUNTS available — **phone Dave on 051-227 2299** for details.

Sinclair ZX81 * NEW RELEASES *

DICTATOR

Another great adventure game from Bug-byte for the 16K ZX81. This time, you are the president of a small state. The object of the game is to avoid revolution, escape assassination attempts, and maintain your popularity, while managing the secret police and army, and maintaining a secure economy. This is a very complex simulation, utilising the whole 16K, and the cassette comes with an 8-page booklet giving full instructions and hints on how to survive.

Can you stand up to the pressures of life as a dictator, and prevent unrest from spreading before it's too late? **PRICE £9.00**

CONSTELLATION

Turn your ZX81 into a telescope! This program will produce a simulation of the night sky as seen from any chosen point on earth at any time this century. You can point your "telescope" in any direction, move it up, down, left and right, zoom in or pull out, and display the stars by magnitude or constellation. **PRICE £8.00**

OTHER ZX81 SOFTWARE FROM OUR RANGE:-

ZXAS assembler	**£5.00**	INVADERS	**£4.00**
ZXDB disassembler/debug	**£6.50**	3D OXO & MARS RESCUE	**£4.50**
MULTIFILE filing system	**£17.50**	VIEWTEXT	**£7.00**
THE DAMSEL & THE BEAST	**£6.50**		
STARTREK	**£5.00**	NOTE: all require RAM pack	

ACCESS/BARCLAYCARD orders accepted on 051-227 2642 (24 hrs)

BUG-BYTE SOFTWARE

98-100 THE ALBANY, OLD HALL STREET, LIVERPOOL L3 9EP

ACORN ATOM SOFTWARE

Acorn Atom INVADERS

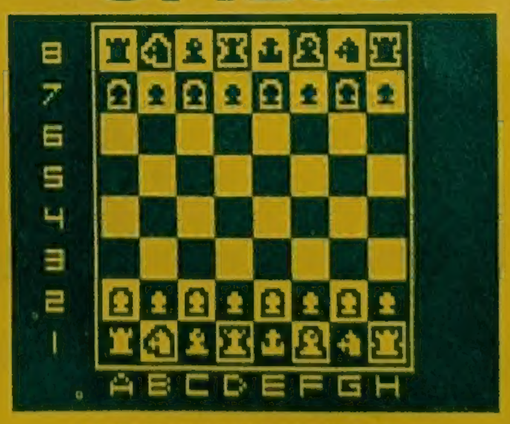
"The best excuse to expand your Atom. It is a superb version of the standard game, tough and fast" Your computer Nov '81 12K, graphics mode 4 **ONLY £8.00**

Acorn Atom FRUIT MACHINE

"A brilliant, incredibly fast model with lovely sound & graphics." Your Computer Nov '81

8K, graphics mode 2. **ONLY £4.00**

CHESS



THE PROGRAM YOU'VE BEEN WAITING FOR!

Fantastic machine code chess game for the 12K Atom. Features include: screen (high res. + alphanumerics); many levels of play; castling & en passant; computer plays black or white. Supplied on cassette with instructions. *PRICE ONLY £9.00*

747

FLIGHT SIMULATION PROGRAM FOR THE 12K ATOM

Written for Bug-byte by a 747 pilot. Accurate simulation of a 747's cockpit display (airspeed, altitude, rate of climb, attitude, flaps, etc., and graphic display of horizontal situation and attitude); allows you to guide your craft to the landing strip. On making your final approach the display changes to a high-resolution 3D representation of the runway coming up to meet you. A real test of skill. Finding the runway is quite a challenge — landing safely is even more difficult. If you succeed, you are awarded a skill rating and the chance to take off and try again.

REQUIRES FLOATING POINT ROM **PRICE ONLY £8.00**

GALAXIAN

Fantastic high-resolution (mode 4) arcade game with fast-swooping aliens, excellent sound effects, and high score. 12K

PRICE £8.00

BACKGAMMON (7K, BASIC)

The program draws out a representation of a backgammon board and allows you to play the standard game against the computer. Playing instructions are not included, but if you can't already play the game, there are several books available to teach you, and the Atom makes an ideal, ever willing partner to build up your playing strength against.

Computer responses are rapid (approximately 10 seconds) and the program will not accept illegal moves. Dice throwing is controlled by the computer. **£7.00**

LABYRINTH (12K, F.P., BASIC, GR.Mod2A, sound)

High-resolution colour graphics (also effective in Black & White) make this 30 maze program one of the best versions available. To help you find your way through the bewildering array of corridors of the 30 x 15 cell random maze, you can call up to a 20-map of the maze. In the harder game option, this map shows only the portions of the maze which you have explored.

Your task is to find your way to the treasure room at the centre of the maze and then escape through the exit. To make things more difficult, several monsters (including the dreaded Minotaur) are loose in the maze, and you will have to fight you way past them. The types of monsters present, and their weapons can be altered by the user, if required. **£7.00**

REQUIRES THE FLOATING POINT ROM

LUNAR LANDER (12K, BASIC & s.c., Gr. Mod4)

A highly addictive arcade style program. A rugged lunar landscape is drawn out and you have to attempt to land your craft safely on the flat areas, by varying the thrust of your main & steering rockets. If you succeed, the ship takes off, and you have to try to land it again, under slightly more difficult conditions. This continues, until you have reduced 3 ships to heaps of rubble.

On screen readout of fuel and score. Several skill levels. A record is kept of the high score. If you are a sufficiently expert pilot, you will be rewarded with extra ships. Definitely a cut above the average lunar lander!! **£5.50**

GOLF (8K, F.P., BASIC)

An 18 hole, par 72 course, complete with fairways, rough, bunkers, trees, streams & greens. Skill and careful club selection are required to get round with a good score. To make things more difficult, you have to specify a fault in your game, which the program will reproduce, and your handicap. At the end of the round, the program produces your scorecard for the round.

A highly entertaining program, which is likely to have you up late into the night straining for a par! **£5.00**

REQUIRES THE FLOATING POINT ROM

DON'T FORGET — OUR PRICES INCLUDE VAT & POSTAGE

ALL OUR PRICES ARE INCLUSIVE!

Please cross cheques/POs & make them payable to "Bug-byte". Payment may be made by Access or Barclaycard, and there is a 24hr answering service on 051-227 2642

GENEROUS DISCOUNT AVAILABLE FOR DEALERS — PHONE FOR DETAILS

BUG-BYTE SOFTWARE

98-100 THE ALBANY, OLD HALL STREET, LIVERPOOL L3 9EP